

AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A computer implemented parallel processing method of for performing a logic simulation, comprising:

representing signals on a line over a time period as a bit sequence[,];

evaluating the output of any logic gate gate outputs of logic gates including an evaluation of any inherent delay by a comparison between the comparing bit sequences of the inputs of the logic gate gates to a predetermined series of bit patterns and in which these logic gates whose outputs have changed over the time period are identified during the evaluation of the gate outputs as real gate changes and only these the logic gates having the real gate changes are propagated to the respective fan out gates of these the logic gates [,] having the real gate changes characterised in that the control of the method is carried out:

in an associative memory mechanism (1a, 1b) which stores storing in word form in an associative memory mechanism a history of gate input signals by compiling a hit list register of logic gate state changes and using:

generating an address for each hit in the hit list via a multiple response resolver (7) forming a part of the associative memory mechanism (1a, 1b), which generates an address for each hit, and then scans transfers the transferring results on the hit list to an output register for subsequent use; and

dividing an associative register into separate smaller associative sub-registers, allocating one type of logic gate to each associative sub-register, each of which associative sub-registers has corresponding sub-registers connected thereto, and carrying out gate evaluations and tests in parallel on each associative sub-register.

2. (Currently Amended) [[A]] The method as claimed in claim 1, in which further comprising storing each delay is stored as a delay word in an the associative register memory (1b) forming part of the associative memory mechanism (1a, 1b) in which[[:-]],

wherein the storing step comprises:

the determining a length of the delay word is ascertained; and

if the length of the delay word width exceeds a register word length of the associative register (1b) word width[[:-]], calculating a the number of integer multiples of the register word width length contained within the delay word is calculated as a gate state[[;]], storing the gate state is stored in a further state register (1a); and storing a the remainder from the calculation is stored in the associative register (1b) with these the delay words whose widths lengths did not exceed the associative register word length, width; and

wherein on the count of when a count of the associative register commences commencing[[:-]]:

the state register (1a) is consulted for the delay word entered in the state register and the remainder is ignored for this the respective count of the associative register;

at the end of the count of the associative register, the state register is updated; and the count continues until the remainder represents that the count is still required.

3. (Currently Amended) [[A]] The method as claimed in claim 1, in which further comprising:

segmenting the hit list ~~is-segmented~~ into a plurality of separate smaller hit lists, each smaller hit list being connected to a separate scan register ~~and in which~~; and transmitting in parallel results of each scan register ~~is operated in parallel to transfer the~~ results to the output register.

4. (Cancelled)

5. (Currently Amended) [[A]] The method as claimed in claim 1, ~~in which~~ further comprising storing each line signal to a target logic gate ~~is-stored~~ as a plurality of bits each representing a delay of one time period, ~~the~~ wherein aggregate bits representing the a delay between a signal output to and reception by the target logic gate, and ~~in which~~ the inherent delay of each logic gate is represented in the same manner.

6. (Currently Amended) [[A]] The method as claimed in claim [[4]]1, ~~in which~~ further comprising using each associative sub-register ~~is-used~~ to form a hit list connected to a corresponding separate scan register.

7. (Currently Amended) [[A]] The method as claimed in claim 1, further comprising using more than one associative sub-register ~~when a~~ ~~in which~~ ~~where~~ the number of ~~the~~ one type of logic gate exceeds a predetermined number ~~more than one~~ ~~sub register~~ ~~is used~~.

8. (Currently Amended) [[A]] The method as claimed in claim 3, further comprising controlling in which the scan registers are controlled by exception logic using an OR gate whereby the scan is terminated for each register on the OR gate changing a state thus indicating no further matches.

9. (Currently Amended) [[A]] The method as claimed in claim 8, wherein in which the scan is carried out by sequential sequentially counting through the hit list and the steps are performed of performing the steps of:

 checking if the bit is set indicating a hit;
 if a hit, determining the address effected by that hit;
 storing the address of the hit;
 clearing the bit in the hit list;
 moving to the a next position in the hit list; and
 repeating the above steps until the hit list is cleared.

10. (Currently Amended) [[A]] The method as claimed in claim 1, in which further comprising storing each line signal to a target logic gate is stored as a plurality of bits each representing a delay of one time period, the wherein aggregate bits representing represent the delay between a signal output to and reception by the target logic gate.

11. (Currently Amended) [[A]] The method as claimed in claim 1, further comprising performing in which there is an initialisation initialization phase, in which includes the steps of: inputting specified signal values are inputted to an input circuit including the logic gates;
setting unspecified signal values are set to unknown;
preparing test templates are prepared defining the to define a delay model for each logic gate;
parsing the input circuit is parsed to generate an equivalent circuit consisting of including 2-input logic gates; and
configuring the 2-input logic gates are then configured.

12. (Currently Amended) [[A]] The method as claimed in claim 1, further comprising applying in which a multi-valued logic is applied and in which n bits are used to represent a signal value at any instance in time with n being any arbitrarily chosen logic.

13. (Currently Amended) [[A]] The method as claimed in claim 12, wherein the multi-value logic includes in which an 8-valued logic, is used where 000 represents logic 0, 111 represents logic 1 and 001 to 110 represent represents other arbitrarily defined other signal states.

14. (Currently Amended) [[A]] The method as claimed in claim 12, further comprising storing a in which the sequence of values on a logic gate is stored as a bit pattern forming a unique word in the associative memory mechanism (1a, 1b).

15. (Currently Amended) [[A]] The method as claimed in claim 1, further comprising
storing in which there is stored a record of all values that a logic gate has acquired for the units of delay of ~~the~~ a longest delay in the circuit.

16. (Currently Amended) A parallel processor for a logic event simulation (APPLES) comprising:-

a main processor (30); and

an associative memory mechanism (1a, 1b) including a response resolver (7);

~~characterised in that~~ wherein the associative memory mechanism (1a, 1b) [[comprises:-]]

further comprises:

 a plurality of separate associative sub-registers each for ~~the storage~~ storing in word form of a history of gate input signals for a specified type of logic gate; and

 a plurality of separate additional sub-registers associated with each associative sub-register whereby gate evaluations and tests can be carried out in parallel on each associative sub-register.

17. (Currently Amended) [[A]] The processor as claimed in claim 16, in which
wherein the additional sub-registers comprise an input sub-register, a mask sub-register and a scan sub-register.

18. (Currently Amended) [[A]] The processor as claimed in claim 17, in which
wherein the scan sub-registers are connected to an output register.